Project Full House

Product Description

An innovative application that allows a user to estimate their age of retirement taking different factors into account such as whether they want to own a house, number of children, their salary etc.

Functional Requirements

Landing Page

1. User must be able to select whether they want to login to a profile or create a new profile or use it in Guest Mode.
2. If the user logins to a profile, the system must ask whether the user wants to create a new profile or sign in to an existing profile.
3. User shall be able to login with google
4. If the user creates a new profile, the system must ask for the username and the password.
   3.1. If the username is already in the database, the system must show the message “<user_name> is already taken. Please use another one.”
   3.2. If the password has invalid format, the system must show the message “The entered password has invalid format. Please use a valid format.”
5. If the user already has an account and signs in, the system must ask for the username and the password.
   4.1. If the username is not in the database, the system must show the message “user_name is not in the database.”
   4.2. If the username is valid but the password is invalid, the system must show the message “Invalid password entered.”
6. If the user forgets their password, they can request to change or reset their password.

Main Page

1. User must be able to select their university.
2. User must be able to select whether they are inputting the currently drawn salary or degree.
3. Users shall input their age and system must reject invalid ages outside of range (18,100)
4. User must be able to select their class honours.
5. User must be able to select their degree type out of 280 entries in the Graduate Employment Survey (Dataset) .
6. User must be able to input total car price if they want to own one, system defaults car price to 0
7. User must be able to select the type of house they want.
8. User must be able to input how many children they are planning to have in the future.
9. User must not be able to proceed to view estimated retirement age if the following is missing or invalid:
   - 9.1. Current drawn salary or (Degree Type and Class Honors)
   - 9.2. Current Age
   - 9.3. Total car price if they want to own one
   - 9.4. How many children they are planning to have in the future
   - 9.5. Type of House
   - 9.6. Investment Plans
10. Web Page must display retirement age calculated once valid request submitted
11. User must be able to provide a desired retirement age should they wish to use the suggestion option.

12. User must be able to submit a feedback form.

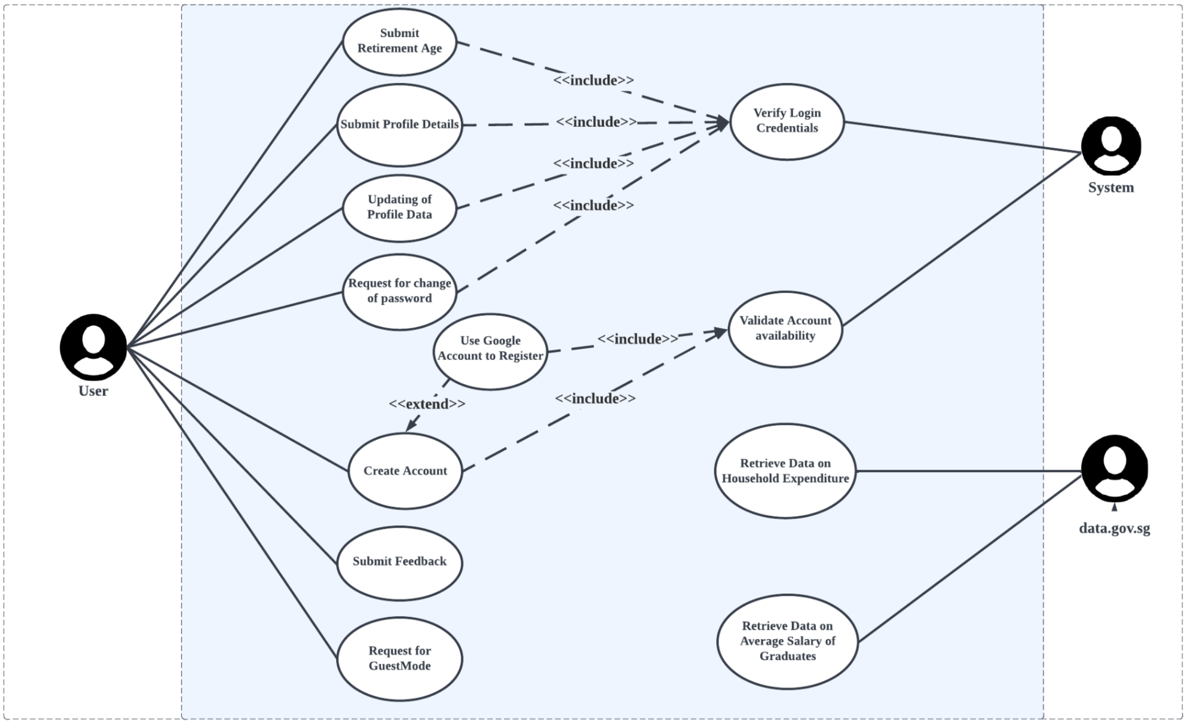
Non-functional Requirements

1. The web application shall load up within 5 seconds
2. The landing page shall be displayed within 1 second of loading the web application.
3. The system must be able to return the display results within 2 seconds.
4. The system must not crash when the user opens the web page.
5. The system must mask the password field in order to prevent any potential shoulder surfing.
6. The system must return relevant feedback to the user if the invalid inputs are detected within 5 seconds.
7. The system must display appropriate messages when processes fail within 10 seconds.

Data Dictionary

Term	**Definition**
Housing	Represents a flat in Singapore. Limited to data of flats that can be found in the dataset from the data.gov.sg website.
User	The user is defined as the end-user of the application.
Class Honors	Class Honors is a type of academic achievement.
Shoulder Surfing	Shoulder Surfing is a type of social engineering technique where another person spies on the user’s when he/she is typing.
User Account	User Account refers to a profile that the user can create and save past results for future reference.
Guest Mode	Anonymous mode that allows the user to access certain functions of the application without logging in.
Data	Data refers to the inputs by the user before calculating the retirement age.
Password	String of characters used to access User Accounts
Submit Feedback	If the user encounters any bugs or errors in the application, they can submit a detailed summary of the event to the server for the review.
Profile Details	User’s profile that contains current drawn salary or degree type, current age, total car price if they want to own one, no. of children they plan to have and the type of housing.
Validate Availability	When creating a new user account, the system checks whether there is an existing user with the same user name to ensure there are no duplicate user names in the database.

Use Case Diagram



Use Case Descriptions

Use Case ID:	1		
Use Case Name:	Verify Login Credentials		
Created By:	Moe	Last Updated By:	Don
Date Created:	27 August 2022	Date Last Updated:	14 September 2022

Actor:	System
Description:	Username and password submitted by user is authenticated on server side
Preconditions:	1. User must be on a stable Wifi/Connection to internet 2. Database must have the User Account registered.
Postconditions:	1. User is able to submit profile details 2. User is able to update profile details
Priority:	High
Frequency of Use:	Whenever user enters the app
Flow of Events:	1. User clicks on Login button on the web application 2. User enters Username or Email 3. User enters Password 4. User clicks on Login button on credentials page 5. System verifies the account by checking for the entry in the Database and matching the password 6. System authenticates the user to login successfully
Alternative Flows:	AF1-S3:System detects credentials fields to be empty 1. Web page displays an error message “Credentials are empty” 2. Return to step 2 AF2-S1:User clicks on Login with google button 1. Return to step 5 AF3-S4:User submits wrong credentials 1. Web page displays an error message “Credentials are invalid, Please re-enter” 2. Return to step 2
Exceptions:	EX-AF2-S1: User is not logged in to google 1. User is brought to gmail login page by system
Includes:	-
Special Requirements:	-
Assumptions:	Database can verify both ways of logging in (google and Username)
Notes and Issues:	-

Use Case ID:	2		
Use Case Name:	Create Account		
Created By:	Don	Last Updated By:	Khant
Date Created:	27 August 2022	Date Last Updated:	14 September 2022

Actor:	User
Description:	Creating a new account
Preconditions:	1. Account must not have already existed. 2. User must have access to the internet.
Postconditions:	1. User will have to enter a verification code sent to their registered email. 2. System registers the details of the user into the database.
Priority:	Medium
Frequency of Use:	1-3 uses per lifetime
Flow of Events:	1. User keys in Username, Password, Confirm-Password, and Email. 2. User clicks on the ‘Register’ button. 3. System checks the availability of the details and cross-checks the password with the Confirm-password field. 4. System will send verification code to the registered email.
Alternative Flows:	AF-S3a: System detected different inputs for Password and Confirm-Password. 1. System will display an error message “Password and Confirm-Password mis-match.” 2. Return to Step 1. AF-S3b: System detected duplicate username and email in the database. 1. System will display the appropriate error message “Email has been taken” or “Username has been taken”. 2. Return to Step 1. AF-S3c: Email does not exist. 1. System will display the appropriate error message “Email does not exist”. 2. Return to Step 1.
Exceptions:	-
Includes:	1. Validate Account Availability
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	3		
Use Case Name:	Request for Password Change		
Created By:	Adrian	Last Updated By:	Don
Date Created:	27 August 2022	Date Last Updated:	14 September 2022

Actor:	User
Description:	Allow user to reset their password.
Preconditions:	1. User must have access to the internet. 2. Database must have the User Account registered.
Postconditions:	1. System successfully updated User's new password in the database.
Priority:	Medium
Frequency of Use:	1-2 per year.
Flow of Events:	1. User navigate to request password reset page. 2. User insert their email and tap on request button. 3. System will verify with database if it matches. 4. System sends an OTP to User's registered email address. 5. User will enter the OTP code and tap the submit button. 6. System will validate the OTP code. 7. User will enter new password and tap save. 8. System will update the new password into the database.
Alternative Flows:	AF-S3: The email provided does not match the database. 1. System will display "Email not registered". 2. Return to step 1. AF-S6: Code entered is invalid. 1. System will display "Invalid OTP code". 2. Return to step 5. AF-S7: User inputs old password into new password. 1. System will display "Cannot reuse old password". 2. Return to step 7.
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	4		
Use Case Name:	Updating of Profile Data		
Created By:	Adrian	Last Updated By:	Adrian
Date Created:	27 August 2022	Date Last Updated:	14 September 2022

Actor:	User
Description:	To allow User to change details in their profile data accordingly.
Preconditions:	1. User must have access to the internet. 2. User must be logged in.
Postconditions:	1. System has successfully updated User’s profile according to user’s submission details.
Priority:	High
Frequency of Use:	0-3 times per year.
Flow of Events:	1. User navigate to user profile page. 2. User updates details accordingly. a. Birthday b. Highest Education/Degree Type c. Class Honors d. Total car price if they want to own one e. How many children they are planning to have in the future f. Type of House g. Investment Plans 3. Details submitted will then be updated to system data.
Alternative Flows:	AF1-S2b: User wants to input salary manually or education is not bachelors 1. User can change the default salary in the field 2. Return to Step 2 AF2-S3: System detects that there are empty fields or incorrect fields. 1. System will display an error message “Please fill up all fields correctly”. 3. Return to Step 2.
Exceptions:	-
Includes:	1. Verify Login Credentials
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	5		
Use Case Name:	Submit Profile Details		
Created By:	Don	Last Updated By:	Khant
Date Created:	27 August 2022	Date Last Updated:	14 September 2022

Actor:	User
Description:	User the submits the profile details into the system.
Preconditions:	1. User has keyed in all relevant information for the website to calculate retirement age. 2. User must be logged in. 3. User must have access to the internet.
Postconditions:	1. System successfully computes the retirement age.
Priority:	High
Frequency of Use:	1 use per lifetime
Flow of Events:	1. User keys in Highest Education, Current Salary, Current Age, No. of Children, Type of House, Total Car Price, Retirement Plans. 2. User clicks the Calculate Retirement Age button. 3. The retirement age will be updated into the profile.
Alternative Flows:	AF-S2: System detects that there are empty or incorrect fields. 2. System will display an error message “Please fill up all fields correctly”. 3. Return to step 1
Exceptions:	-
Includes:	1. Verify Login Credentials
Special Requirements:	-
Assumptions:	All fields can be filled up by User.
Notes and Issues:	-

Use Case ID:	6		
Use Case Name:	Submit Feedback		
Created By:	Khant	Last Updated By:	Tran Que An
Date Created:	27 August 2022	Date Last Updated:	14 September 2022

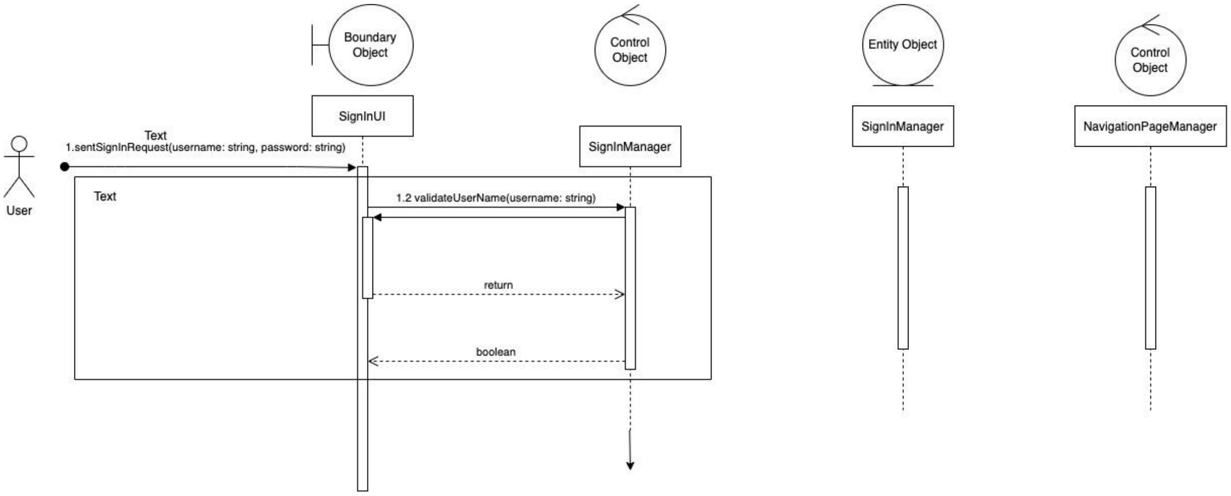
Actor:	User
Description:	Submitting the feedback for the application
Preconditions:	1. User must have access to the internet.
Postconditions:	1. Feedback is successfully submitted for review.
Priority:	Low
Frequency of Use:	1-2 per year.
Flow of Events:	1. User enters the user name, the feedback title, and the summary of the feedback. 2. User clicks on “Submit” button. 3. The feedback details are then sent to the server.
Alternative Flows:	AF-S2: System detects that there are empty fields. 1. System will display error message “Please fill up all empty fields”. 2. Return to step 1
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

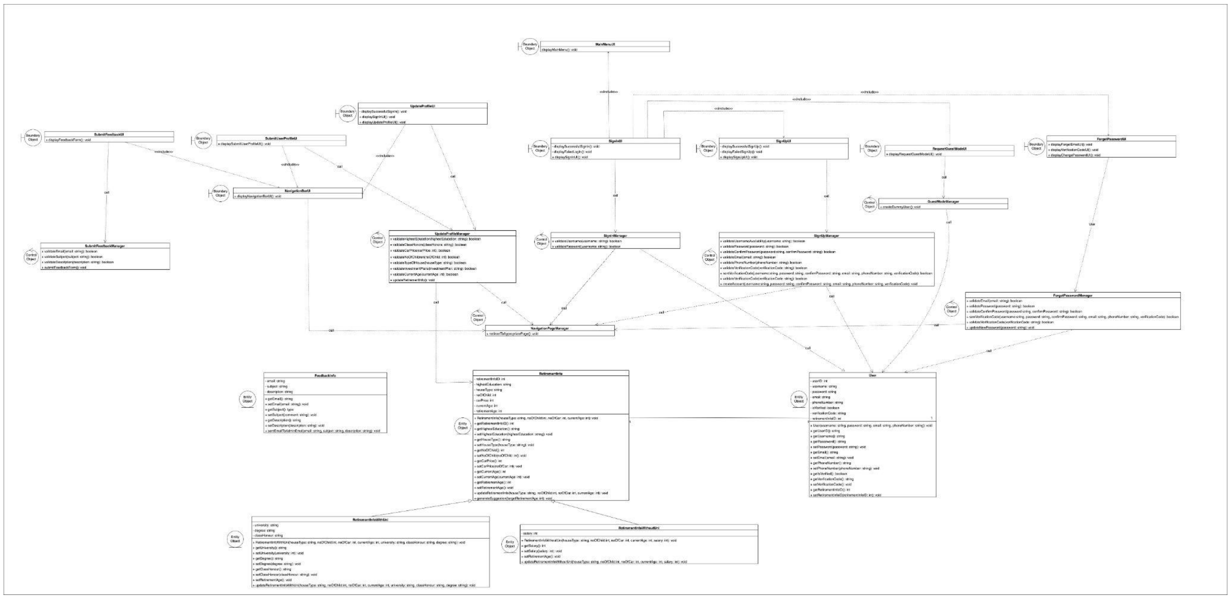
Use Case ID:	7		
Use Case Name:	Request For Guest Mode		
Created By:	Khant	Last Updated By:	Don
Date Created:	27 August 2022	Date Last Updated:	14 September 2022

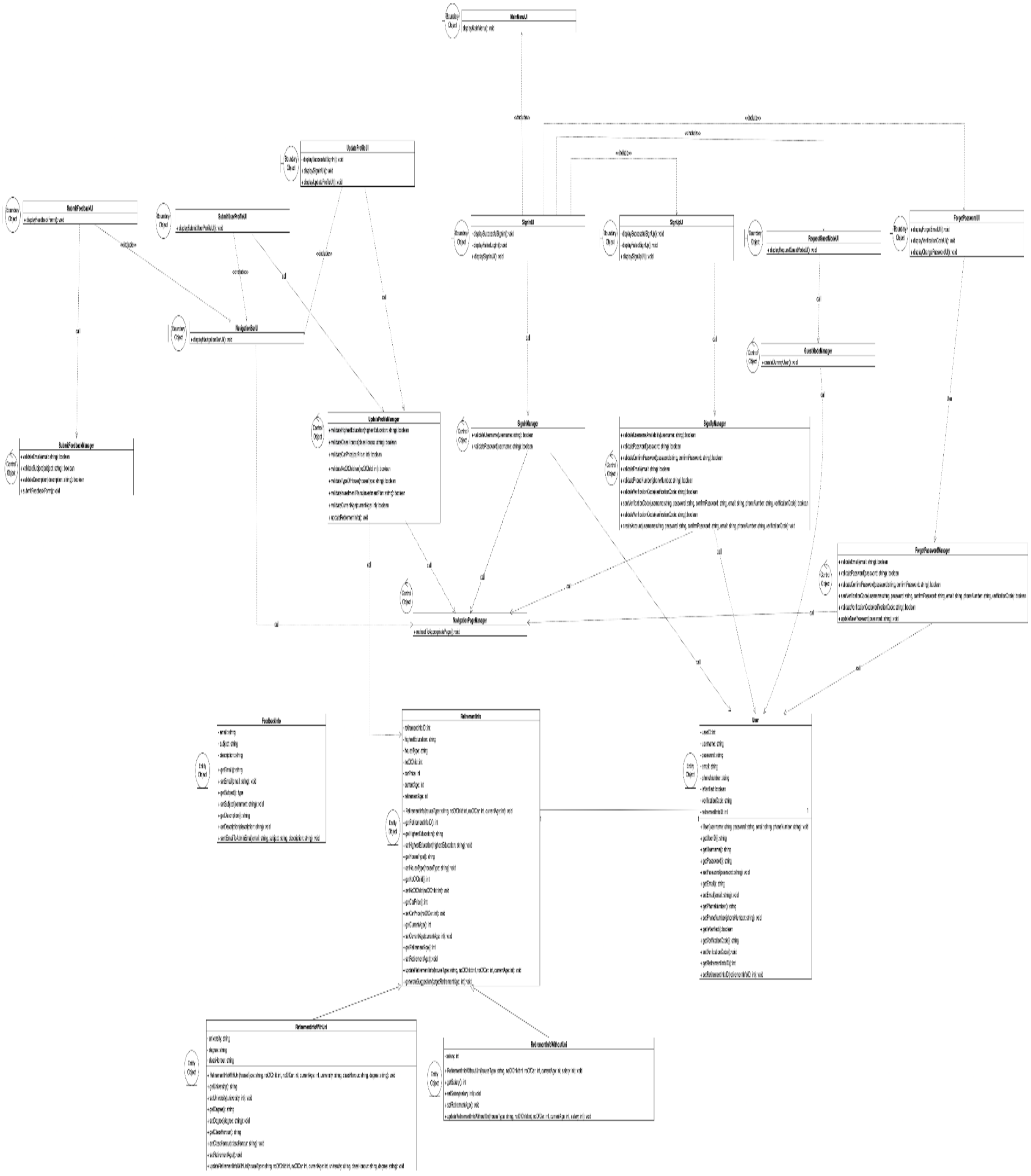
Actor:	User
Description:	Enter guest mode which allows the user to access certain functions in the system without logging in.
Preconditions:	1. User must have access to the internet.
Postconditions:	1. User is able to access to the system under the guest mode.
Priority:	Medium
Frequency of Use:	1 per month
Flow of Events:	1. The user clicks on “Guest Mode” button. 2. User enters the system under Guest Mode.
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	8		
Use Case Name:	Enter retirement age		
Created By:	Don	Last Updated By:	
Date Created:	14 September 2022	Date Last Updated:	

Actor:	User
Description:	User enters their desired retirement age for system to suggest changes to lifestyle to meet the desired age.
Preconditions:	1. User must have access to the internet. 2. User must have a desired retirement age. 3. Details in the user profile are filled.
Postconditions:	1. User is provided with suggestions on how they can meet their entered retirement age if their current parameters are not sufficient.
Priority:	High
Frequency of Use:	1-3 times per year
Flow of Events:	1. User enters desired retirement age 2. User presses submit button 3. Suggestions box will appear and display accordingly 4. User selects Update. 5. Profile in system is updated.
Alternative Flows:	AF-S2: System detects that there are empty fields. 1. System will display an error message “Please fill up all empty fields”. 2. Return to step 1 AF-S3: User selects close. 1. System box will close and suggestion will not update the system.
Exceptions:	-
Includes:	1. Verify Login Credentials 2. Submit Profile Details
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-


Boundary Object
Control Object
Entity Object
Control Object
SignInUI
SignInManager
SignInManager
NavigationPageManager
User
Text
1.sentSignInRequest(username: string, password: string)
Text
1.2 validateUserName(username: string)
return
boolean


Boundary Object
MainMenuUI
displayMainMenu(): void
<<include>>
UpdateProfileUI
- displaySuccessfulSignIn(): void
+ displaySignInUI(): void
+ displayUpdateProfileUI(): void
SubmitFeedbackUI
+ displayFeedbackForm(): void
SubmitUserProfileUI
+ displaySubmitUserProfileUI(): void
NavigationBarUI
+ displayNavigationBarUI(): void
SignInUI
- displaySuccessfulSignIn(): void
- displayFailedLogIn(): void
+ displaySignInUI(): void
SignUpUI
- displaySuccessfulSignUp(): void
- displayFailedSignUp(): void
+ displaySignUpUI(): void
RequestGuestModeUI
+ displayRequestGuestModeUI(): void
ForgetPasswordUI
+ displayForgetEmailUI(): void
+ displayVerificationCodeUI(): void
+ displayChangePasswordUI(): void
call
use
Control Object
SubmitFeedbackManager
+ validateEmail(email: string): boolean
+ validateSubject(subject: string): boolean
+ validateDescription(description: string): boolean
+ submitFeedbackForm(): void
UpdateProfileManager
+ validateHighestEducation(highestEducation: string): boolean
+ validateClassHonors(classHonors: string): boolean
+ validateCarPrice(carPrice: int): boolean
+ validateNoOfChildren(noOfChild: int): boolean
+ validateTypeOfHouse(houseType: string): boolean
+ validateInvestmentPlans(investmentPlan: string): boolean
+ validateCurrentAge(currentAge: int): boolean
+ updateRetirementInfo(): void
SignInManager
+ validateUsername(username: string): boolean
+ validatePassword(username: string): boolean
GuestModeManager
+ createDummyUser(): void
SignUpManager
+ validateUsernameAvailability(username: string): boolean
+ validatePassword(password: string): boolean
+ validateConfirmPassword(password:string, confirmPassword: string): boolean
+ validateEmail(email: string): boolean
+ validatePhoneNumber(phoneNumber: string): boolean
+ validateVerificationCode(verificationCode: string): boolean
+ sentVerificationCode(username:string, password: string, confirmPassword: string, email: string, phoneNumber: string, verificationCode): boolean
+ validateVerificationCode(verificationCode: string): boolean
+ createAccount(username:string, password: string, confirmPassword: string, email: string, phoneNumber: string, verificationCode): void
ForgetPasswordManager
+ validateEmail(email: string): boolean
+ validatePassword(password: string): boolean
+ validateConfirmPassword(password:string, confirmPassword: string): boolean
+ sentVerificationCode(username:string, password: string, confirmPassword: string, email: string, phoneNumber: string, verificationCode): boolean
+ validateVerificationCode(verificationCode: string): boolean
+ updateNewPassword(password: string): void
NavigationPageManager
+ redirectToAppropriatePage(): void
Entity Object
FeedbackInfo
- email: string
- subject: string
- description: string
+ getEmail(): string
+ setEmail(email: string): void
+ getSubject(): type
+ setSubject(comment: string): void
+ getDescription(): string
+ setDescription(description: string): void
+ sentEmailToAdminEmail(email: string, subject: string, description: string): void
RetirementInfo
- retirementInfoID: int
- highestEducation: string
- houseType: string
- noOfChild: int
- carPrice: int
- currentAge: int
- retirementAge: int
+ RetirementInfo(houseType: string, noOfChild:int, noOfCar: int, currentAge: int): void
+ getRetirementInfoID(): int
+ getHighestEducation(): string
+ setHighestEducation(highestEducation: string): void
+ getHouseType(): string
+ setHouseType(houseType: string): void
+ getNoOfChild(): int
+ setNoOfChild(noOfChild: int): void
+ getCarPrice(): int
+ setCarPrice(noOfCar: int): void
+ getCurrentAge(): int
+ setCurrentAge(currentAge: int): void
+ getRetirementAge(): int
+ setRetirementAge(): void
+ updateRetirementInfo(houseType: string, noOfChild:int, noOfCar: int, currentAge: int): void
+ generateSuggestion(targetRetirementAge: int): void
User
- userID: int
- username: string
- password: string
- email: string
- phoneNumber: string
- isVerified: boolean
- verificationCode: string
- retirementInfoID: int
1
+ User(username: string, password: string, email: string, phoneNumber: string): void
+ getUserID(): string
+ getUsername(): string
+ getPassword(): string
+ setPassword(password: string): void
+ getEmail(): string
+ setEmail(email: string): void
+ getPhoneNumber(): string
+ setPhoneNumber(phoneNumber: string): void
+ getIsVerified(): boolean
+ getVerificationCode(): string
+ setVerificationCode(): void
+ getRetirementInfoID(): int
+ setRetirementInfoID(retirementInfoID: int): void
RetirementInfoWithUni
- university: string
- degree: string
- classHonour: string
+ RetirementInfoWithUni(houseType: string, noOfChild:int, noOfCar: int, currentAge: int, university: string, classHonour: string, degree: string): void
+ getUniversity(): string
+ setUniversity(university: int): void
+ getDegree(): string
+ setDegree(degree: string): void
+ getClassHonour(): string
+ setClassHonour(classHonour: string): void
+ setRetirementAge(): void
+ updateRetirementInfoWithUni(houseType: string, noOfChild:int, noOfCar: int, currentAge: int, university: string, classHonour: string, degree: string): void
RetirementInfoWithoutUni
- salary: int
+ RetirementInfoWithoutUni(houseType: string, noOfChild:int, noOfCar: int, currentAge: int, salary: int): void
+ getSalary(): int
+ setSalary(salary: int): void
+ setRetirementAge(): void
+ updateRetirementInfoWithoutUni(houseType: string, noOfChild:int, noOfCar: int, currentAge: int, salary: int): void


Boundary Object
MainMenuUI
displayMainMenu(): void
<<include>>
<<include>>
<<include>>
<<include>>
Boundary Object
UpdateProfileUI
- displaySuccessfulSignIn(): void
+ displaySignInUI(): void
+ displayUpdateProfileUI(): void
Boundary Object
SubmitFeedbackUI
+ displayFeedbackForm(): void
Boundary Object
SubmitUserProfileUI
+ displaySubmitUserProfileUI(): void
<<include>>
<<include>>
<<include>>
call
Boundary Object
NavigationBarUI
+ displayNavigationBarUI(): void
Boundary Object
SignInUI
- displaySuccessfulSignIn(): void
- displayFailedLogIn(): void
+ displaySignInUI(): void
Boundary Object
SignUpUI
- displaySuccessfulSignUp(): void
- displayFailedSignUp(): void
+ displaySignUpUI(): void
Boundary Object
RequestGuestModeUI
+ displayRequestGuestModeUI(): void
Boundary Object
ForgetPasswordUI
+ displayForgetEmailUI(): void
+ displayVerificationCodeUI(): void
+ displayChangePasswordUI(): void
Control Object
GuestModeManager
+ createDummyUser(): void
Use
Control Object
SubmitFeedbackManager
+ validateEmail(email: string): boolean
+ validateSubject(subject: string): boolean
+ validateDescription(description: string): boolean
+ submitFeedbackForm(): void
Control Object
UpdateProfileManager
+ validateHighestEducation(highestEducation: string): boolean
+ validateClassHonours(classHonours: string): boolean
+ validateCarPrice(carPrice: int): boolean
+ validateNoOfChildren(noOfChild: int): boolean
+ validateTypeOfHouse(houseType: string): boolean
+ validateInvestmentPlans(investmentPlan: string): boolean
+ validateCurrentAge(currentAge: int): boolean
+ updateRetirementInfo(): void
Control Object
SignInManager
+ validateUsername(username: string): boolean
+ validatePassword(username: string): boolean
Control Object
SignUpManager
+ validateUsernameAvailability(username: string): boolean
+ validatePassword(password: string): boolean
+ validateConfirmPassword(password:string, confirmPassword: string): boolean
+ validateEmail(email: string): boolean
+ validatePhoneNumber(phoneNumber: string): boolean
+ validateVerificationCode(verificationCode: string): boolean
+ sentVerificationCode(username:string, password: string, confirmPassword: string, email: string, phoneNumber: string, verificationCode): boolean
+ validateVerificationCode(verificationCode: string): boolean
+ createAccount(username:string, password: string, confirmPassword: string, email: string, phoneNumber: string, verificationCode): void
Control Object
ForgetPasswordManager
+ validateEmail(email: string): boolean
+ validatePassword(password: string): boolean
+ validateConfirmPassword(password:string, confirmPassword: string): boolean
+ sentVerificationCode(username:string, password: string, confirmPassword: string, email: string, phoneNumber: string, verificationCode): boolean
+ validateVerificationCode(verificationCode: string): boolean
+ updateNewPassword(password: string): void
Control Object
NavigationPageManager
+ redirectToAppropriatePage(): void
call
Entity Object
FeedbackInfo
- email: string
- subject: string
- description: string
+ getEmail(): string
+ setEmail(email: string): void
+ getSubject(): type
+ setSubject(comment: string): void
+ getDescription(): string
+ setDescription(description: string): void
+ sentEmailToAdminEmail(email: string, subject: string, description: string): void
Entity Object
RetirementInfo
- retirementInfoID: int
- highestEducation: string
- houseType: string
- noOfChild: int
- carPrice: int
- currentAge: int
- retirementAge: int
+ RetirementInfo(houseType: string, noOfChild:int, noOfCar: int, currentAge: int): void
+ getRetirementInfoID(): int
+ getHighestEducation(): string
+ setHighestEducation(highestEducation: string): void
+ getHouseType(): string
+ setHouseType(houseType: string): void
+ getNoOfChild(): int
+ setNoOfChild(noOfChild: int): void
+ getCarPrice(): int
+ setCarPrice(noOfCar: int): void
+ getCurrentAge(): int
+ setCurrentAge(currentAge: int): void
+ getRetirementAge(): int
+ setRetirementAge(): void
+ updateRetirementInfo(houseType: string, noOfChild:int, noOfCar: int, currentAge: int): void
+ generateSuggestion(targetRetirementAge: int): void
1
1
1
Entity Object
User
- userID: int
- username: string
- password: string
- email: string
- phoneNumber: string
- isVerified: boolean
- verificationCode: string
- retirementInfoID: int
+ User(username: string, password: string, email: string, phoneNumber: string): void
+ getUserID(): string
+ getUsername(): string
+ getPassword(): string
+ setPassword(password: string): void
+ getEmail(): string
+ setEmail(email: string): void
+ getPhoneNumber(): string
+ setPhoneNumber(phoneNumber: string): void
+ getIsVerified(): boolean
+ getVerificationCode(): string
+ setVerificationCode(): void
+ getRetirementInfoID(): int
+ setRetirementInfoID(retirementInfoID: int): void
Entity Object
RetirementInfoWithUni
- university: string
- degree: string
- classHonour: string
+ RetirementInfoWithUni(houseType: string, noOfChild:int, noOfCar: int, currentAge: int, university: string, classHonour: string, degree: string): void
+ getUniversity(): string
+ setUniversity(university: int): void
+ getDegree(): string
+ setDegree(degree: string): void
+ getClassHonour(): string
+ setClassHonour(classHonour: string): void
+ setRetirementAge(): void
+ updateRetirementInfoWithUni(houseType: string, noOfChild:int, noOfCar: int, currentAge: int, university: string, classHonour: string, degree: string): void
Entity Object
RetirementInfoWithoutUni
- salary: int
+ RetirementInfoWithoutUni(houseType: string, noOfChild:int, noOfCar: int, currentAge: int, salary: int): void
+ getSalary(): int
+ setSalary(salary: int): void
+ setRetirementAge(): void
+ updateRetirementInfoWithoutUni(houseType: string, noOfChild:int, noOfCar: int, currentAge: int, salary: int): void